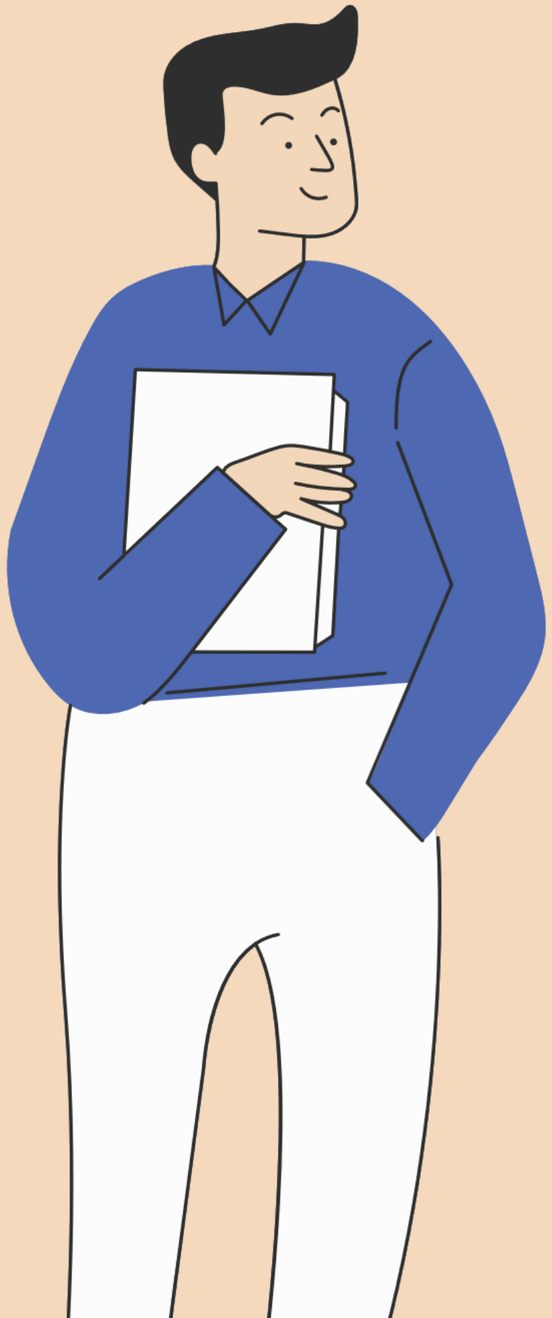


8WEEKSQLCHALLENGE.COM

CASE STUDY #1

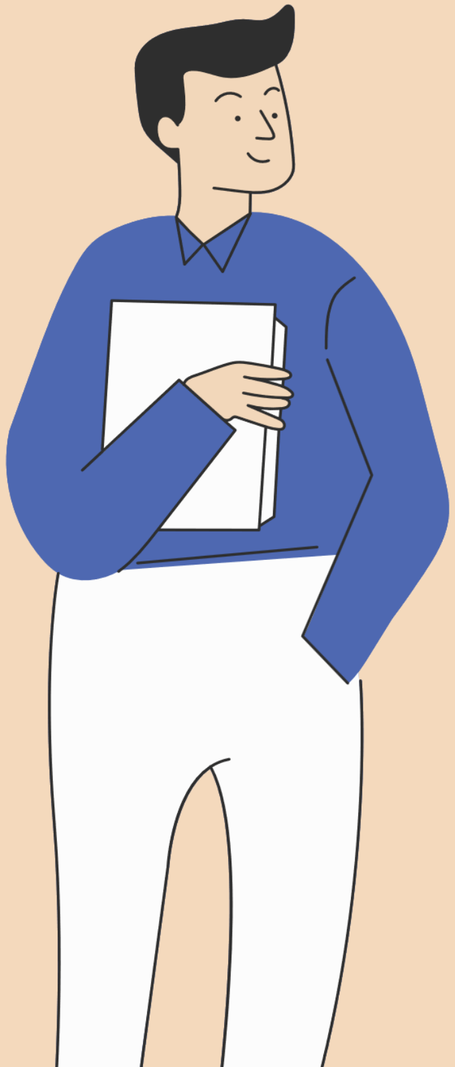
THE TASTE OF SUCCESS

DATAWITHDANNY.COM

AGENDA

1. Case Study Introduction
2. Problem Statement
3. Table Relationship
4. Question and Solution

Case Study Introduction



Danny seriously loves Japanese food so in the beginning of 2021, he decides to embark upon a risky venture and opens up a cute little restaurant that sells his 3 favorite foods: sushi, curry and ramen.

Danny's Diner is in need of your assistance to help the restaurant stay afloat – the restaurant has captured some very basic data from their few months of operation but have no idea how to use their data to help them run the business.

Problem Statement

Danny wants to use the data to answer a few simple questions about his customers, especially about their

- visiting patterns,
- how much money they’ve spent, and
- which menu items are their favorite.

Having this deeper connection with his customers will help him deliver a better and more personalized experience for his loyal customers.

He plans on using these insights to help him decide whether he should expand the existing customer loyalty program – additionally he needs help to generate some basic datasets so his team can easily inspect the data without needing to use SQL.

The data set contains the following 3 tables which you may refer to the relationship diagram below to understand the connection.

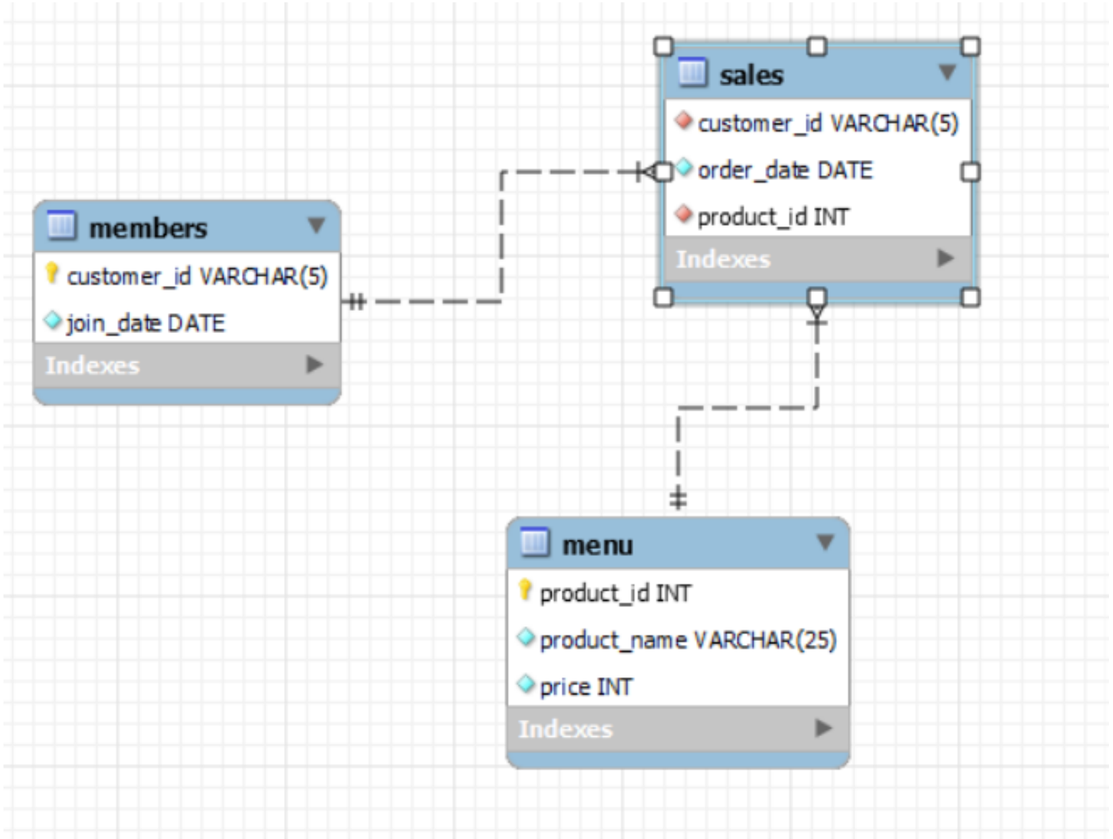
sales
members
menu

Table View & Relationship

customer_id	join_date
A	2021-01-07
B	2021-01-09

customer_id	order_date	product_id
A	2021-01-01	1
A	2021-01-01	2
A	2021-01-07	2
A	2021-01-10	3
A	2021-01-11	3
A	2021-01-11	3
B	2021-01-01	2
B	2021-01-02	2
B	2021-01-04	1
B	2021-01-11	1
B	2021-01-16	3
B	2021-02-01	3
C	2021-01-01	3
C	2021-01-01	3
C	2021-01-07	3

product_id	product_name	price
1	sushi	10
2	curry	15
3	ramen	12



Question and Solution

1. What is the total amount each customer spent at the restaurant?

```sql
-- 1st Approch
SELECT
DISTINCT s.customer_id,
SUM(m.price) OVER (PARTITION BY s.customer_id) AS Total_amount
FROM sales AS s
LEFT JOIN menu AS m
    ON s.product_id = m.product_id;
```

```sql
-- 2nd  Approch --
SELECT
s.customer_id,
SUM(m.price) AS Total_amount
FROM sales AS s
LEFT JOIN menu AS m
    ON s.product_id = m.product_id
GROUP BY s.customer_id;
```

customer_id	Total_amount
A	76
B	74
C	36

- **Customer A spent $76.**
- **Customer B spent $74.**
- **Customer C spent $36.**

2. How many days has each customer visited the restaurant ?

```sql
SELECT
customer_id AS Customer,
COUNT(DISTINCT order_date) AS no_of_days_visited
FROM sales
GROUP BY customer_id;
```

Customer	no_of_days_visited
A	4
B	6
C	2

- Customer A visited 4 times.
- Customer B visited 6 times.
- Customer C visited 2 times.

3. What was the first item from the menu purchased by each customer?

```sql
WITH first_purchase_item AS(
SELECT
s.customer_id,
order_date,
m.product_name,
DENSE_RANK() OVER (PARTITION BY s.customer_id ORDER BY order_date ASC ) AS rnk
FROM sales AS s
LEFT JOIN menu AS m
    ON s.product_id = m.product_id)
Select customer_id, product_name
FROM first_purchase_item
WHERE rnk = 1
GROUP BY customer_id,product_name;
```

customer_id	product_name
A	sushi
A	curry
B	curry
C	ramen

- **Customer A's first order are curry and sushi.**
- **Customer B's first order is curry.**
- **Customer C's first order is ramen.**

4. What is the most purchased item on the menu and how many times was it purchased by all customers?

```sql
SELECT
DISTINCT m.product_name,
COUNT(*) OVER (PARTITION BY s.product_id) AS no_of_times_purchased
FROM sales AS s
LEFT JOIN menu AS m
    ON s.product_id = m.product_id
ORDER BY no_of_times_purchased DESC
LIMIT 1 ;
```

product_name	no_of_times_purchased
ramen	8

- **The most purchased item on the menu is ramen which is 8 times. Yummy!!!!**

5. Which item was the most popular for each customer ?

```sql
WITH most_purchase AS(
SELECT
m.product_name,
s.customer_id,
COUNT(m.product_id) AS no_of_times_purchased,
DENSE_RANK() OVER (PARTITION BY s.customer_id ORDER BY COUNT(s.customer_id) DESC) AS rnk
FROM sales AS s
LEFT JOIN menu AS m
    ON  m.product_id = s.product_id
GROUP BY s.customer_id, m.product_name)
SELECT customer_id, product_name, no_of_times_purchased FROM most_purchase WHERE rnk = 1;
```

Customer	product_name	no_of_times_purchased
A	ramen	3
B	curry	2
B	sushi	2
B	ramen	2
C	ramen	3

- **Customer A and C's favorite item is ramen.**
- **Customer B enjoys all items on the menu. He/she is a true foodie.**

6. Which item was purchased first by the customer after they became a member?

```
WITH CTE1 AS (
SELECT s.*, m.product_name, mem.join_date
FROM sales AS s
LEFT JOIN members AS mem
    ON s.customer_id = mem.customer_id
LEFT JOIN menu AS m
        ON s.product_id = m.product_id
WHERE s.order_date > mem.join_date),
CTE2 AS( SELECT customer_id, product_name, order_date, join_date,
DENSE_RANK() OVER (PARTITION BY customer_id ORDER BY order_date ASC) AS rnk
FROM CTE1)
SELECT DISTINCT customer_id , product_name
FROM CTE2 WHERE rnk = 1;
```

customer_id	product_name
A	ramen
B	sushi

- Customer A's first order as a member is ramen.
- Customer B's first order as a member is sushi.

7. Which item was purchased just before the customer became a member?

```sql
WITH purchase_before_member AS(
SELECT m.customer_id, s.product_id,
ROW_NUMBER() OVER (PARTITION BY m.customer_id ORDER BY s.order_date DESC) AS rnk
FROM members AS m
INNER JOIN sales AS s
    ON m.customer_id = s.customer_id
    AND s.order_date < m.join_date)
SELECT pbm.customer_id, mn.product_name
FROM purchase_before_member AS pbm
INNER JOIN menu AS mn
    ON pbm.product_id = mn.product_id
WHERE pbm.rnk = 1
ORDER BY pbm.customer_id;
```

Customer	Product_Name
A	sushi
B	sushi

- Both customers' last order before becoming members are sushi. That must have been a really good sushi!!

8. What is the total items and amount spent for each member before they became a member?

```
WITH CTE1 AS (
SELECT mem.customer_id, s.product_id
FROM members AS mem
LEFT JOIN sales AS s
    ON mem.customer_id = s.customer_id
    AND s.order_date < mem.join_date)
SELECT  CTE1.customer_id AS Customer,
COUNT(CTE1.product_id) AS Total_items, SUM(m.price) AS Total_amount_spent
FROM CTE1
LEFT JOIN menu AS m
ON CTE1.product_id = m.product_id
GROUP BY CTE1.customer_id;
```

Customer	Total_items	Total_amount_spent
A	2	25
B	3	40

Before becoming members,

- **Customer A spent $25 on 2 items.**
- **Customer B spent $40 on 3 items.**

9. If each $1 spent equates to 10 points and sushi has a 2x points multiplier – how many points would each customer have?

```sql
WITH points AS(
SELECT *,
CASE WHEN product_id = 1 THEN price * (2 * 10)
ELSE price * 10 END AS product_point
FROM menu)
SELECT DISTINCT s.customer_id AS Customer,
SUM(p.product_point) OVER (PARTITION BY s.customer_id) AS Total_points
FROM SALES AS s
LEFT JOIN points p
    ON s.product_id = p.product_id;
```

Customer	Total_points
A	860
B	940
C	360

The total points for Customers A, B and C are $860, $940 and $360.

10. In the first week after a customer joins the program (including their join date) they earn 2x points on all items, not just sushi — how many points do customer A and B have at the end of January?

```sql
WITH date_cte AS(
SELECT customer_id, join_date,
CAST(join_date + 6 AS DATE) AS Valid_date,
CAST('2021-01-31' AS DATE) AS Last_date
FROM members)
SELECT s.customer_id,
SUM(CASE WHEN s.order_date BETWEEN d.join_date AND d.Valid_date
    THEN 2 * 10 * m.price
    WHEN s.order_date NOT BETWEEN d.join_date AND d.Valid_date
    AND m.product_name LIKE 'sushi' THEN 2 * 10 * m.price
    ELSE 10 * m.price END) AS Total_points
    FROM sales AS s
INNER JOIN date_cte AS d
    ON s.customer_id = d.customer_id
    ND s.order_date <= d.Last_date
INNER JOIN menu AS m
    ON s.product_id = m.product_id
GROUP BY s.customer_id;
```

Customer	Total_points
A	1370
B	820

Customer A has 1,370 points.
Customer B has 820 points.

Bonus Questions

Join All The Things

Recreate the table with: customer_id, order_date, product_name, price, member (Y/N)

```sql
SELECT s.customer_id AS Customer, s.order_date AS Order_Date,
mn.product_name AS Product_Name, mn.price AS Price,
CASE
WHEN s.order_date < m.join_date THEN 'N'
WHEN s.order_date > m.join_date THEN 'Y'
ELSE 'N' END AS Membership
FROM sales AS s
LEFT JOIN members AS m
    ON s.customer_id = m.customer_id
LEFT JOIN menu AS mn
    ON s.product_id = mn.product_id
ORDER BY s.customer_id, s.order_date;
```

Customer	Order_Date	Product_Name	Price	Membership
A	2021-01-01	sushi	10	N
A	2021-01-01	curry	15	N
A	2021-01-07	curry	15	N
A	2021-01-10	ramen	12	Y
A	2021-01-11	ramen	12	Y
A	2021-01-11	ramen	12	Y
B	2021-01-01	curry	15	N
B	2021-01-02	curry	15	N
B	2021-01-04	sushi	10	N
B	2021-01-11	sushi	10	Y
B	2021-01-16	ramen	12	Y
B	2021-02-01	ramen	12	Y
C	2021-01-01	ramen	12	N
C	2021-01-01	ramen	12	N
C	2021-01-07	ramen	12	N

Rank All The Things

Danny also requires further information about the ranking of customer products, but he purposely does not need the ranking for non-member purchases so he expects null ranking values for the records when customers are not yet part of the loyalty program.

```sql
WITH CTE1 AS (
SELECT s.customer_id AS Customer, s.order_date AS Order_Date,
mn.product_name AS Product_Name, mn.price AS Price,
CASE
WHEN s.order_date < m.join_date THEN 'N'
WHEN s.order_date >= m.join_date THEN 'Y'
ELSE 'N' END AS Membership
FROM sales AS s
LEFT JOIN members AS m
    ON s.customer_id = m.customer_id
INNER JOIN menu AS mn
    ON s.product_id = mn.product_id
ORDER BY s.customer_id, s.order_date)
SELECT *,
CASE
WHEN Membership = 'N' THEN NULL
ELSE RANK() OVER (PARTITION BY customer_id, Membership ORDER BY order_date)
END AS Ranking
FROM CTE1;
```

Customer	Order_Date	Product_Name	Price	Membership	Ranking
A	2021-01-01	sushi	10	N	NULL
A	2021-01-01	curry	15	N	NULL
A	2021-01-07	curry	15	Y	1
A	2021-01-10	ramen	12	Y	2
A	2021-01-11	ramen	12	Y	3
A	2021-01-11	ramen	12	Y	3
B	2021-01-01	curry	15	N	NULL
B	2021-01-02	curry	15	N	NULL
B	2021-01-04	sushi	10	N	NULL
B	2021-01-11	sushi	10	Y	1
B	2021-01-16	ramen	12	Y	2
B	2021-02-01	ramen	12	Y	3
C	2021-01-01	ramen	12	N	NULL
C	2021-01-01	ramen	12	N	NULL
C	2021-01-07	ramen	12	N	NULL


THANK
YOU!